Objectives

The main objectives of this project are to:

1. Enhance the image quality for better analysis.
2. Detect and highlight the clock face.
3. Identify the clock hands.
4. Calculate the time based on the positions of the clock hands.

Introduction

Analog clocks are a common means of timekeeping, and interpreting the time from an image of an analog clock can be a challenging task. This project aims to use computer vision and image processing techniques to automate this process. The workflow involves several steps, including preprocessing the image, detecting the clock center, identifying the clock hands, and calculating the time.

Clock detection and time recognition from an image is a complex task involving multiple stages of image processing. The main steps include enhancing the image quality, detecting edges and contours, isolating the clock face, and finally determining the positions of the clock hands to calculate the time. This process involves various image processing techniques such as histogram equalization, edge detection, noise removal, morphological transformations, and angle calculations.

Necessary Libraries:

- OpenCV
- Numpy
- Matplotlib
- Skimage
- Math
- Tkinter
- PIL

Working Procedure

1. Preprocessing

Preprocessing is a critical step to enhance the quality of the image and prepare it for further analysis. The preprocessing steps include:

a. Reading the Image

The image of the analog clock is read using OpenCV.

Analog Clock Images:

b. Gamma Correction

Gamma correction is applied to adjust the brightness of the image. This step ensures that the image has a uniform brightness, making it easier to detect features.

Formula for Gamma Correction:

$$I' = \left(\frac{I}{255}\right)^{\left(\frac{1}{\gamma}\right)} * 255$$

Image after gamma correction:

c. Conversion to Grayscale

The image is converted to grayscale to simplify the processing steps that follow. Grayscale images require less computational power and are easier to work with.

Grayscale image:

d. Blurring and Thresholding

The image undergoes Gaussian blurring to reduce noise and smooth out variations, which helps in minimizing irrelevant details. This step prepares the image for better processing by making significant features, like edges, more prominent. Next, adaptive thresholding is applied to convert the blurred grayscale image into a binary image. This binary transformation highlights the edges and contours, making it easier to detect specific features such as the hands and numbers on the clock.

Formula for Gaussian Blur:

$$G(x,y) = \frac{1}{2\pi\sigma} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

- G(x,y): The weight of the kernel at position (x,y).
- σ: The standard deviation of the Gaussian distribution, which determines the extent of the blur.
- x and y: The coordinates of a point in the kernel relative to the center.

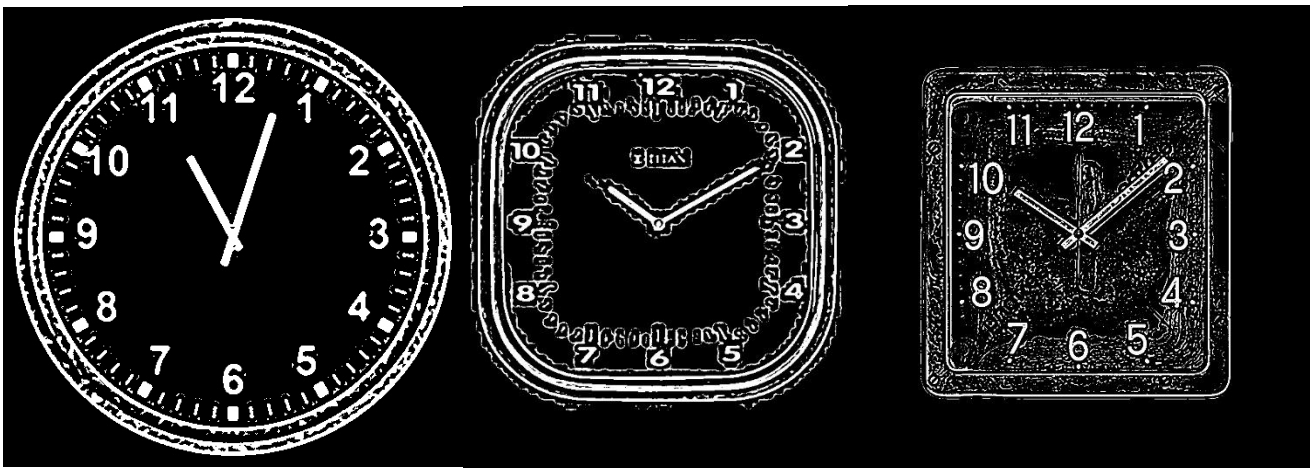
The kernel is applied to each pixel of the image, where the pixel value is replaced by the weighted sum of its neighbors based on this Gaussian function.

Formula for Adaptive Thresholding:

$$T(x,y) = \frac{1}{N} \sum_{i,j \in neighborhood} G(i,j) * I(x+i, y+j) - C$$

- *T(x,y): The threshold value at pixel (x,y).*
- *N: The total number of pixels in the neighborhood.*
- *G(i,j): The Gaussian weight for the pixel at (i,j) relative to the center of the neighborhood.*
- *I(x+i,y+j): The intensity of the pixel at (x+i,y+j) in the image.*
- *C: A constant subtracted from the mean or weighted sum to fine-tune the thresholding.*

Binary image after adaptive thresholding:



e. Dilation and Noise Removal

Dilation is applied to strengthen the features in the binary image. The dilation of A by B is defined by

$$A \oplus B = \bigcup_{b \in B} A_b$$

Where A_b is the translation of A by b. Further noise removal steps are performed to eliminate small, irrelevant details that could interfere with the clock hand detection.

After the dilation, a series of median filters is applied to the image to reduce salt-and-pepper noise by replacing each pixel with median value of its neighborhood.

Equation for Median Filter:

$$I'(x,y) = medianI(x+i, y+j)$$

where (i,j) represents the neighborhood around (x,y).

This is followed by a bilateral filtering, which reduces noise while preserving edges by averaging pixels based on both their spatial proximity and intensity similarity.

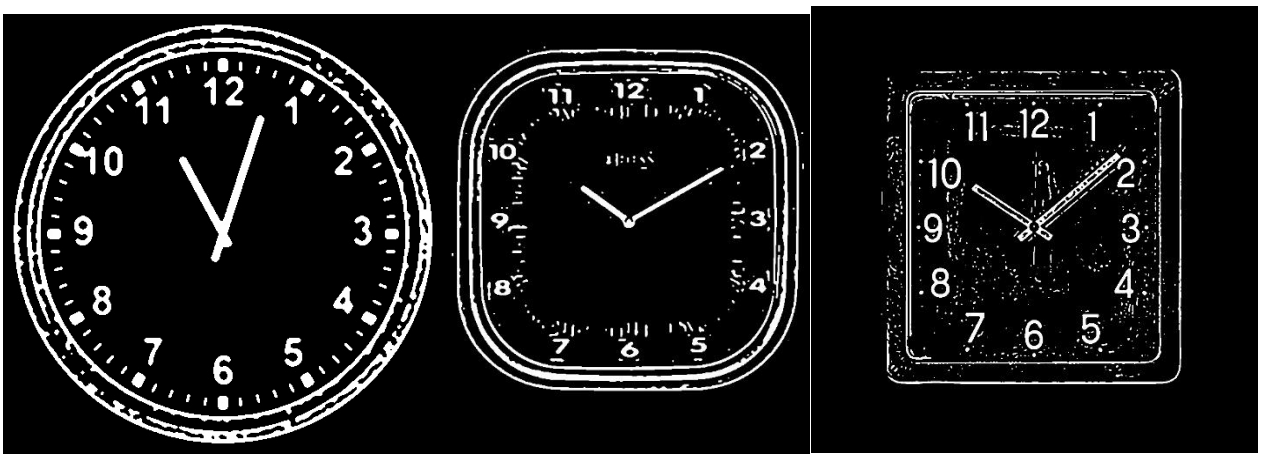
Equation of Bilateral Filter:

The bilateral filter output $I'(x,y)$ is computed as:

$$[I'(x,y) = \frac{1}{W_p}\sum_{i,j} I(i,j) \cdot \exp\left(-\frac{(i-x)^2 + (j-y)^2}{2\sigma_d^2}\right) \cdot \exp\left(-\frac{|I(i,j) - I(x,y)|^2}{2\sigma_r^2}\right)]$$

where Wp is a normalization factor, σ_d controls the spatial distance, and σ_r controls the intensity difference.

Image after dilation and noise removal:



f. Skeletonization

Skeletonization reduces the shapes in the processed image to their minimal skeletal form using the 'morphology.skeletonize' function, which helps in detecting the clock hands by representing them as thin lines. The skeleton is then converted to an 8-bit format, followed by a morphological closing to connect gaps and enhance continuity. Finally, small components with fewer than 100 pixels are removed to focus on the significant structures.

The morphology.skeletonize function reduces the foreground objects in a binary image to their thinnest possible form without breaking their connectivity. It does this by iteratively removing pixels from the object's boundary while ensuring that the overall shape and topology are preserved. The process continues until no more pixels can be removed without altering the object's structure, resulting in a single-pixel-wide skeleton.

Skeletonization Equation

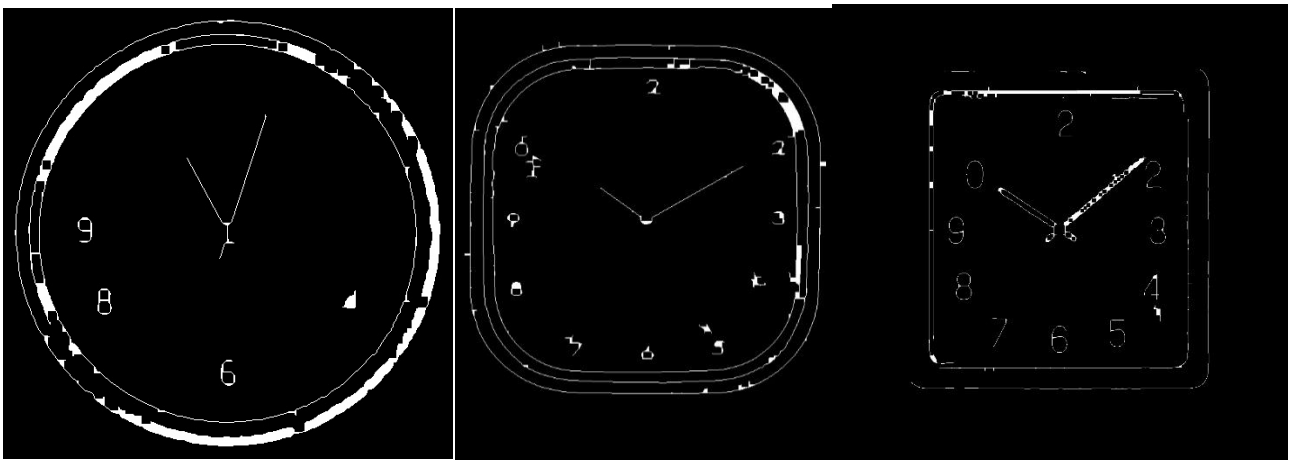
Skeletonization can be represented mathematically as:

$$\text{Skeleton}(A) = \bigcup_{i=0}^{n} (A \ominus iB) - \big((A \ominus iB) \circ B\big)$$

]

where A is the binary image, B is the structuring element, $\ominus$ denotes erosion, $\circ$ denotes opening, and n is the number of iterations.

Skeletonized image:



2. Center Detection

The center of the clock is detected by finding the contour with the maximum area, which is assumed to be the clock face. The highest diameter is also calculated from the max contour. If the contour is unconnected, the center of the image is used as the clock center and the highest diameter is assumed the 5/6 of the image height.

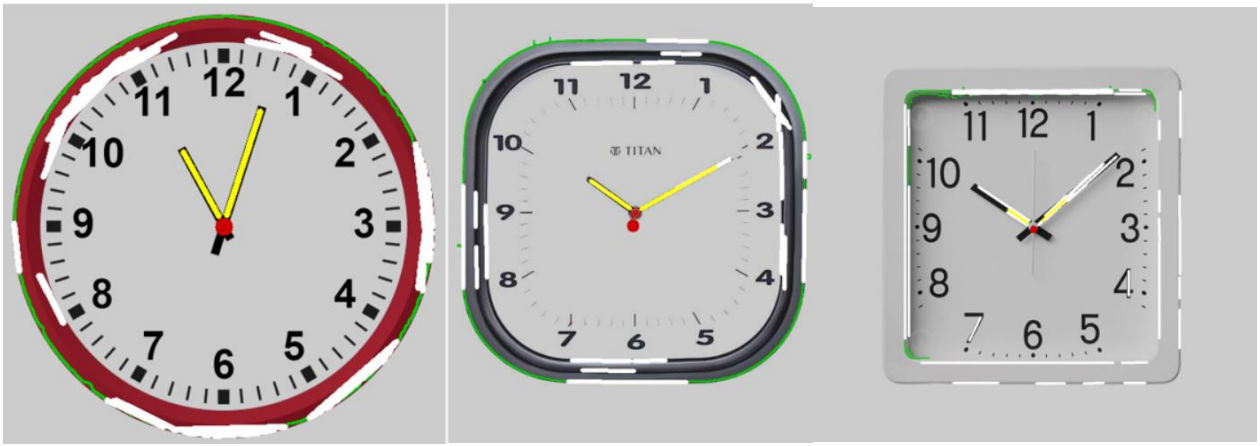
Contour of the clock:

Contour Detection with center detected:

3. Hand Detection

The clock hands are detected using the Hough Line Transform. This technique identifies straight lines in the skeletonized image. The detected lines are filtered based on their proximity to the clock center.

Detected clock hands:



Algorithm for Clock Time Detection

1. Start.
2. Detect lines in the image.
3. If the number of lines is 0:
   - Output: "Image too complex to detect the clock".
4. If the number of lines is 1:
   - Calculate the length of the line.
   - If the length is greater than half the clock's diameter:
     - Output: "Time is 6:00".
   - Else:
     - Output: "Time is 12:00".
5. If the number of lines is greater than 2:
   - Calculate slopes of all lines.
   - Cluster lines into two groups based on slopes.
   - Calculate the average line for each group.
   - Identify the shorter line as the hour hand and the longer line as the minute hand.
   - Calculate the angles of the hour and minute hands relative to the 12 o'clock and 3 o'clock positions.
   - Determine the time based on these angles.
   - Output: "Time".
6. End.

4. Time Calculation

The time is calculated based on the angles formed by the detected clock hands with respect to the 12 o'clock position.

Mathematical Formulas:

We have,

- ✓ the center of the clock: (cx,cy)
- ✓ the mean point of hour hand's detected coordinates: (x_{hour_hand},y_{hour_hand})
- ✓ the mean point of minutes hand's detected coordinates: (x_{min_hand},$y_{,min\_hand}$)

For the 12 o'clock position:

$$(x_{12}, y_{12}) = (cx, cy - 60)$$

For the 3 o'clock position:

$$(x\_\{3\}, y\_\{3\}) = (cx + 60, cy)$$

Hour hand angle relative to the 12 o'clock position:

$$\theta_{\text{hour\_12}} = \arctan\left(\frac{y_{\text{hour\_hand}} - y_{12}}{x_{\text{hour\_hand}} - x_{12}}\right)$$

Hour hand angle relative to the 3 o'clock position:

$$\theta_{\text{hour\_3}} = \arctan\left(\frac{y_{\text{hour\_hand}} - y_{12}}{x_{\text{hour\_hand}} - x_{12}}\right)$$

Minute hand angle relative to the 12 o'clock position:

$$\theta_{\text{minute\_12}} = \arctan\left(\frac{y_{\text{minute\_hand}} - y_{12}}{x_{\text{minute\_hand}} - x_{12}}\right)$$

Minute hand angle relative to the 3 o'clock position:

$$\theta_{\text{minute\_3}} = \arctan\left(\frac{y_{\text{minute\_hand}} - y_{3}}{x_{\text{minute\_hand}} - x_{3}}\right)$$

Adjusted Hour Angle:

$$Hour\ Angle = \begin{cases} 360° - \theta_{hour_{12}} & if\ \theta_{hour_3} > 90° \\ \theta_{hour_{12}} & otherwise \end{cases}$$

Adjusted Minute Angle:

$$our\ Angle = \begin{cases} 360° - \theta_{minute\_12} & if\ \theta_{minute\_3} > 90° \\ \theta_{minute\_12} & otherwise \end{cases}$$

Minutes Calculation:

$$Minute = \lfloor \frac{\text{Minute Angle}}{6} \rfloor$$

Hours Calculation:

$$Hour = \left\lfloor \frac{\text{Hour Angle}}{30} \right\rfloor$$

If Hour = 0, then Hour = 12

Conclusion

In conclusion, the implementation of image processing techniques in interpreting time from analog clock images has proven effective and reliable. By systematically enhancing image quality, detecting the clock center, identifying clock hands, and calculating their positions using mathematical formulas, the system achieves precise time recognition. This approach highlights the capability of image processing to automate tasks traditionally reliant on human perception, demonstrating its potential in various applications where accurate time detection is critical.

Moreover, the robustness of the system lies in its ability to handle diverse clock designs and conditions through adaptive algorithms and precise angle calculations. By circumventing the need for complex computer vision methods and focusing solely on image processing, the project underscores the efficiency and accessibility of this approach. Moving forward, advancements in image processing technology promise further enhancements in accuracy and speed, paving the way for broader applications in fields ranging from automated surveillance to industrial automation.

Theoretical Aspects and Calculations

Image Enhancement

1. **Gamma Correction:**
    - Adjust the brightness and contrast using a gamma correction technique.

```
pseudo
Copy code
Create gamma correction lookup table
Apply gamma correction using the lookup table
```

Noise Removal

1. **Median and Bilateral Filtering:**
    - Apply median filters of different kernel sizes to reduce noise.
    - Apply bilateral filter for further noise reduction while preserving edges.

```
pseudo
Copy code
Apply median blur with different kernel sizes
Apply bilateral filter
```

Edge and Contour Detection

1. **Morphological Transformations:**

- Convert the image to grayscale.
- Apply Gaussian blur and adaptive thresholding.
- Use opening and closing operations to remove small white regions and black holes.

```
pseudo
Copy code
Convert to grayscale
Apply Gaussian blur
Apply adaptive thresholding
Perform morphological opening and closing
```

2. **Skeletonization and Component Analysis:**
   - Skeletonize the binary image.
   - Perform connected component analysis to remove small components.

```
pseudo
Copy code
Skeletonize the image
Perform connected component analysis
Remove small components
```

Clock Face Detection

1. **Contour Analysis:**
   - Find contours in the skeletonized image.
   - Identify the largest contour, assumed to be the clock face.
   - Calculate the center and diameter of the clock face.

```
pseudo
Copy code
Find contours
Identify the largest contour
Calculate the center and diameter of the largest contour
```

Clock Hand Detection

1. **Hough Line Transform:**
   - Use Hough Line Transform to detect lines in the skeletonized image.
   - Filter lines based on proximity to the clock center.
   - Classify lines into hour and minute hands based on their lengths and angles.

```
pseudo
Copy code
Detect lines using Hough Line Transform
Filter lines near the clock center
Classify lines into hour and minute hands
```

Time Calculation

1. **Angle Calculation:**
   - Calculate the angles of the clock hands relative to the 12 o'clock position.
   - Convert the angles to time.

```
pseudo
Copy code
Calculate angles of clock hands
Convert angles to time
```

Limitations

1. **Second Hand:**
   - The presence of a second hand can greatly reduce the accuracy of time estimation.
2. **Complex Backgrounds:**
   - The algorithm may struggle with images having complex backgrounds or multiple objects.
3. **Low Contrast:**
   - Low contrast between the clock hands and the clock face can affect detection accuracy.
4. **Distorted Images:**
   - Images with distortions or reflections may lead to incorrect detection.
5. **Hands Overlapping:**
   - Overlapping clock hands can complicate the angle calculation and time estimation.

Exception Handling

1. **Invalid Image Path:**
   - Handle file not found or invalid image path errors.

```
pseudo
Copy code
Try to load image
Catch file not found exception
Display error message
```

2. **Zero Contours Detected:**
   - Handle cases where no contours are detected in the image.

```
pseudo
Copy code
If no contours found
Display error message
```

3. **Division by Zero:**
   - Handle division by zero errors in angle calculations.

```
pseudo
Copy code
Try to calculate angle
Catch division by zero exception
Set angle to default value
```

4. **Invalid Line Detection:**
   - Handle cases where no valid lines are detected for clock hands.

```
pseudo
Copy code
If no valid lines detected
Display error message
```

Pseudocode Summary

```
pseudo
Copy code
Load image
Convert image to LAB and enhance using CLAHE
Apply gamma correction
Remove noise using median and bilateral filtering
Convert to grayscale and apply adaptive thresholding
Perform morphological transformations (opening and closing)
Skeletonize the image and remove small components
Find contours and identify the clock face
Calculate center and diameter of the clock face
Detect lines using Hough Line Transform
Filter lines near the clock center
Classify lines into hour and minute hands
Calculate angles of clock hands relative to 12 o'clock position
Convert angles to time
Display results
Handle exceptions (invalid image path, zero contours, division by zero, invalid line
detection)
```